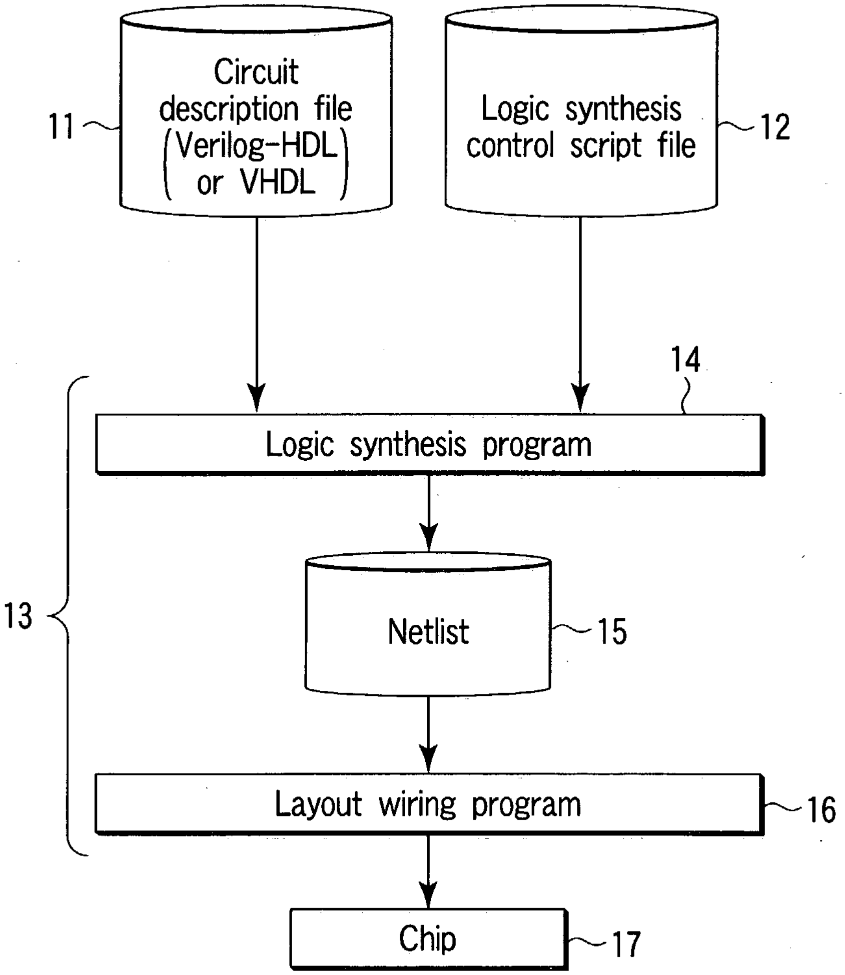

Circuit description file (Verilog-HDL or VHDL)
11
Logic synthesis control script file
12
14
Logic synthesis program
13
Netlist
15
Layout wiring program
16
Chip
17

FIG. 1 PRIOR ART

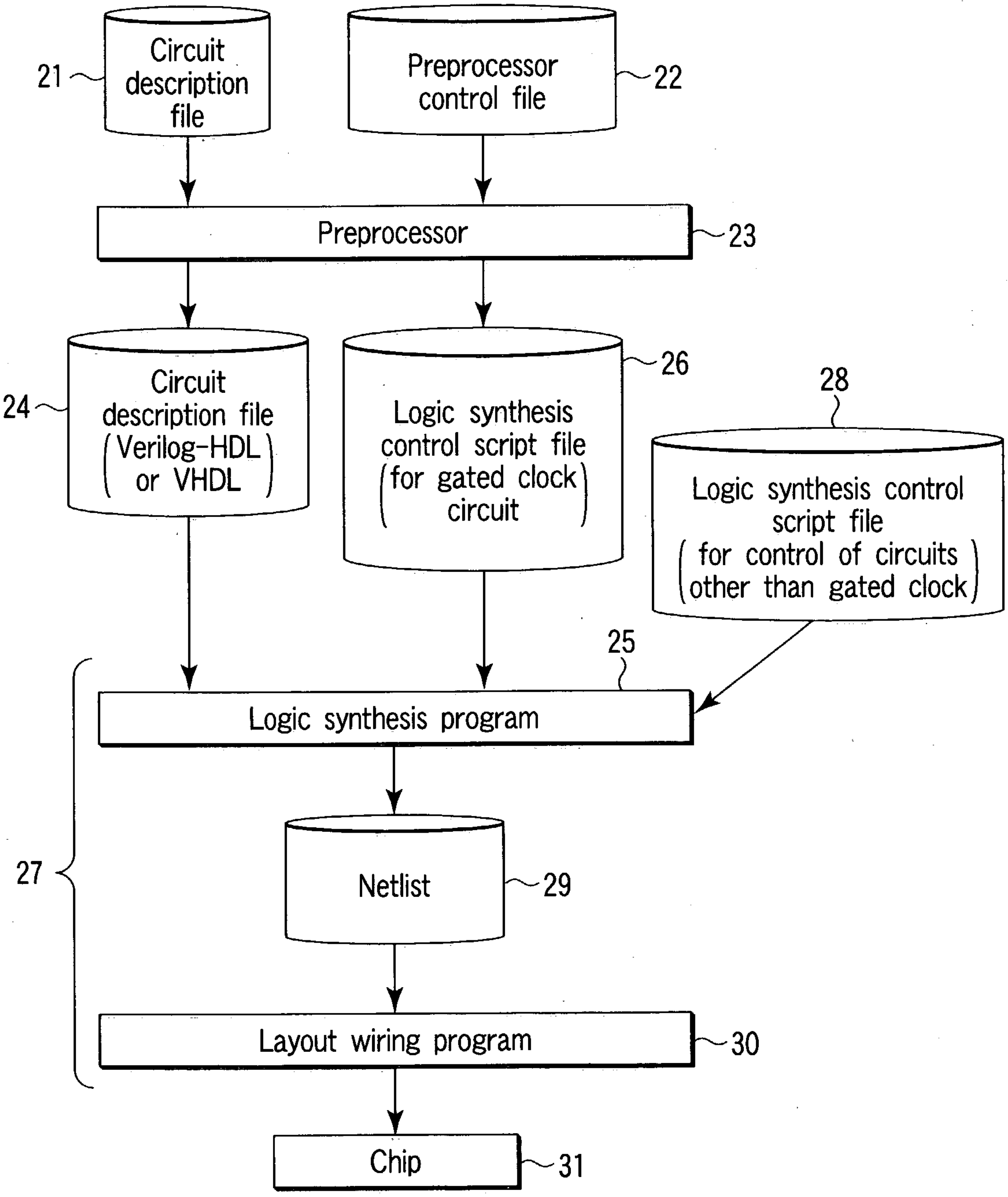
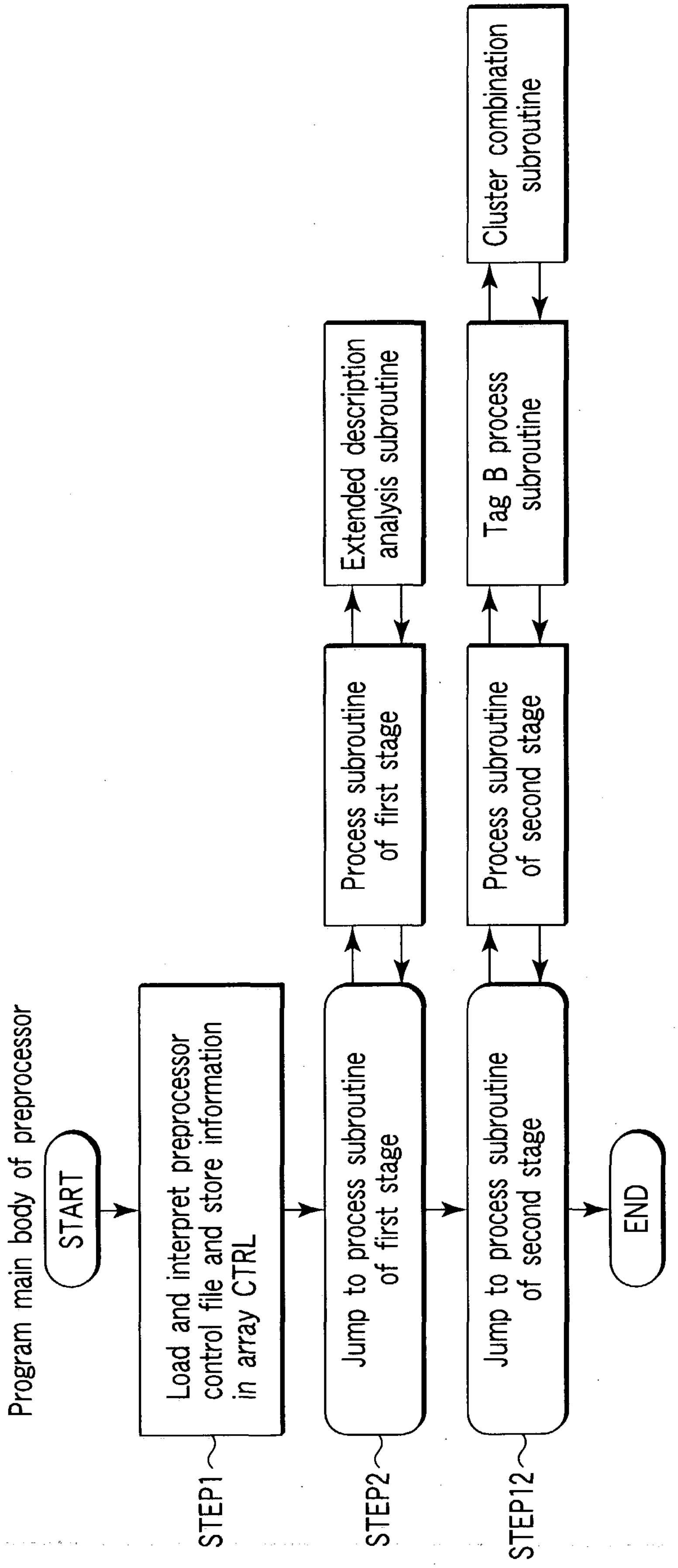

21
Circuit description file
Preprocessor control file
22
Preprocessor
23
24
Circuit description file (Verilog-HDL or VHDL)
Logic synthesis control script file (for gated clock circuit)
26
28
Logic synthesis control script file (for control of circuits other than gated clock)
25
Logic synthesis program
27
Netlist
29
Layout wiring program
30
Chip
31

FIG. 2


Program main body of preprocessor
START
STEP1
Load and interpret preprocessor control file and store information in array CTRL
STEP2
Jump to process subroutine of first stage
Process subroutine of first stage
Extended description analysis subroutine
STEP12
Jump to process subroutine of second stage
Process subroutine of second stage
Tag B process subroutine
Cluster combination subroutine
END

F I G. 3

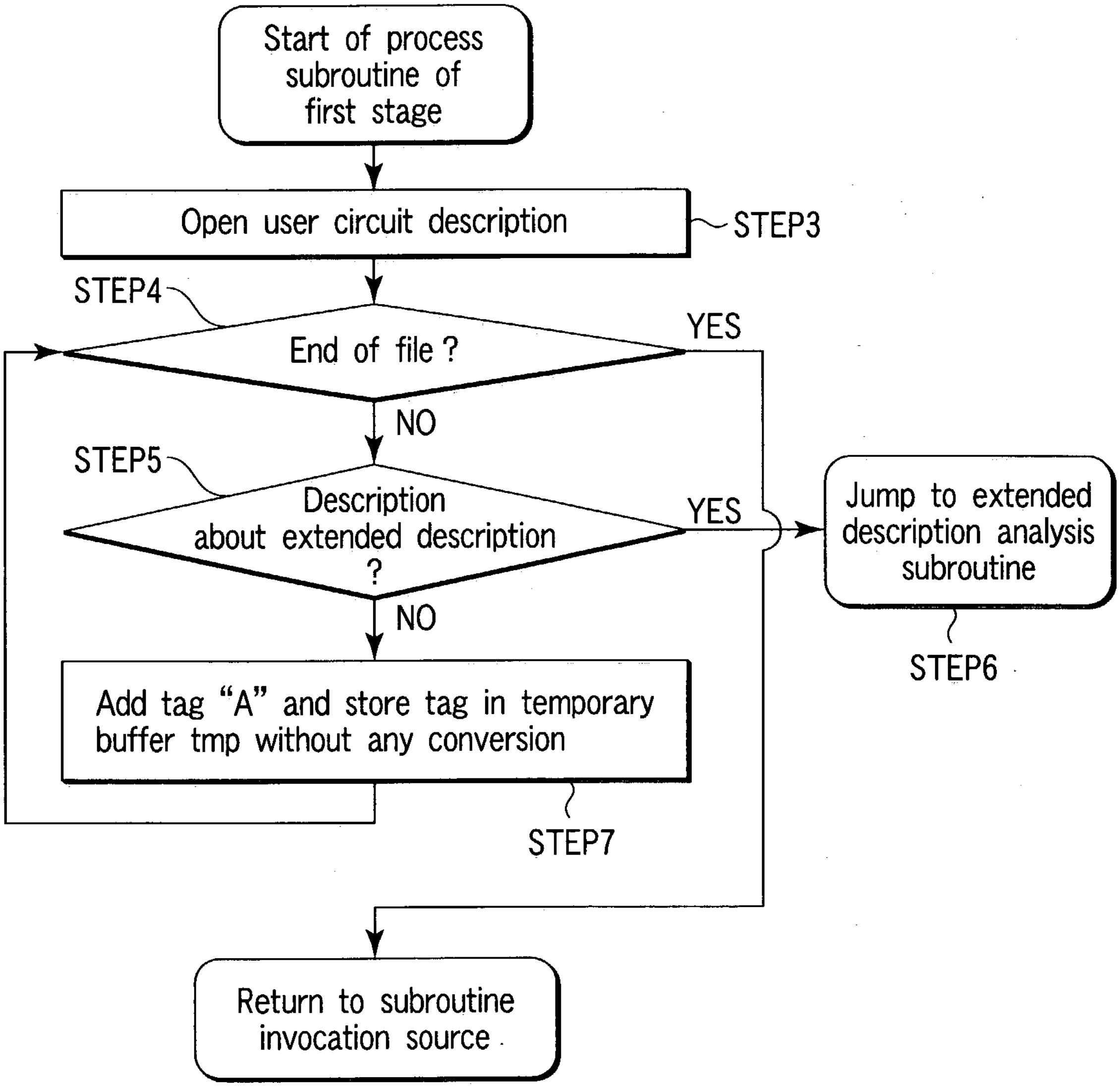

Start of process subroutine of first stage
Open user circuit description
STEP3
STEP4
End of file ?
YES
NO
STEP5
Description about extended description ?
YES
Jump to extended description analysis subroutine
STEP6
NO
Add tag "A" and store tag in temporary buffer tmp without any conversion
STEP7
Return to subroutine invocation source

FIG.4

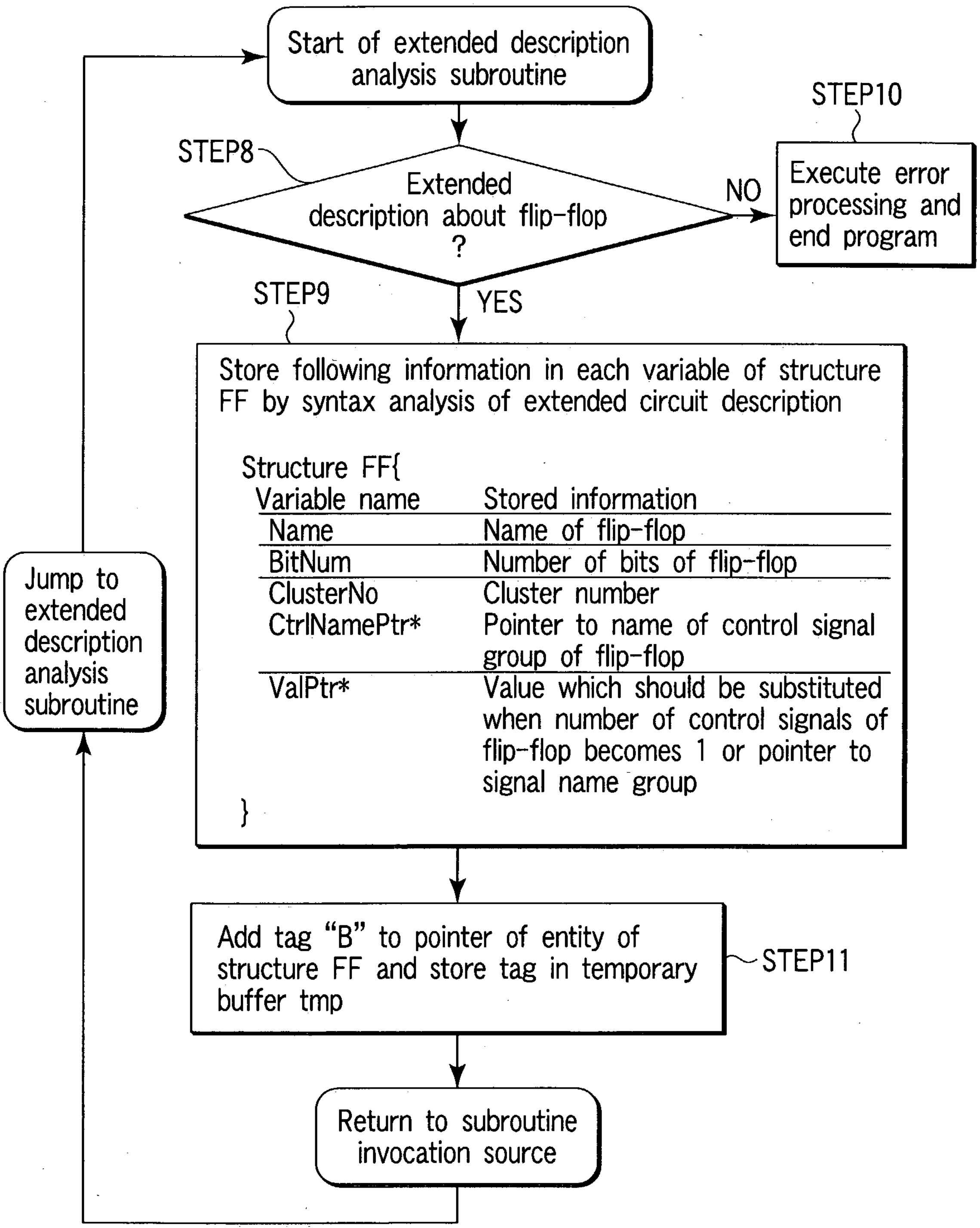

Start of extended description analysis subroutine
STEP8
Extended description about flip-flop ?
NO
STEP10
Execute error processing and end program
YES
STEP9
Store following information in each variable of structure FF by syntax analysis of extended circuit description
Structure FF{
Variable name
Stored information
Name
Name of flip-flop
BitNum
Number of bits of flip-flop
ClusterNo
Cluster number
CtrlNamePtr*
Pointer to name of control signal group of flip-flop
ValPtr*
Value which should be substituted when number of control signals of flip-flop becomes 1 or pointer to signal name group
}
Add tag "B" to pointer of entity of structure FF and store tag in temporary buffer tmp
STEP11
Return to subroutine invocation source
Jump to extended description analysis subroutine

FIG.5

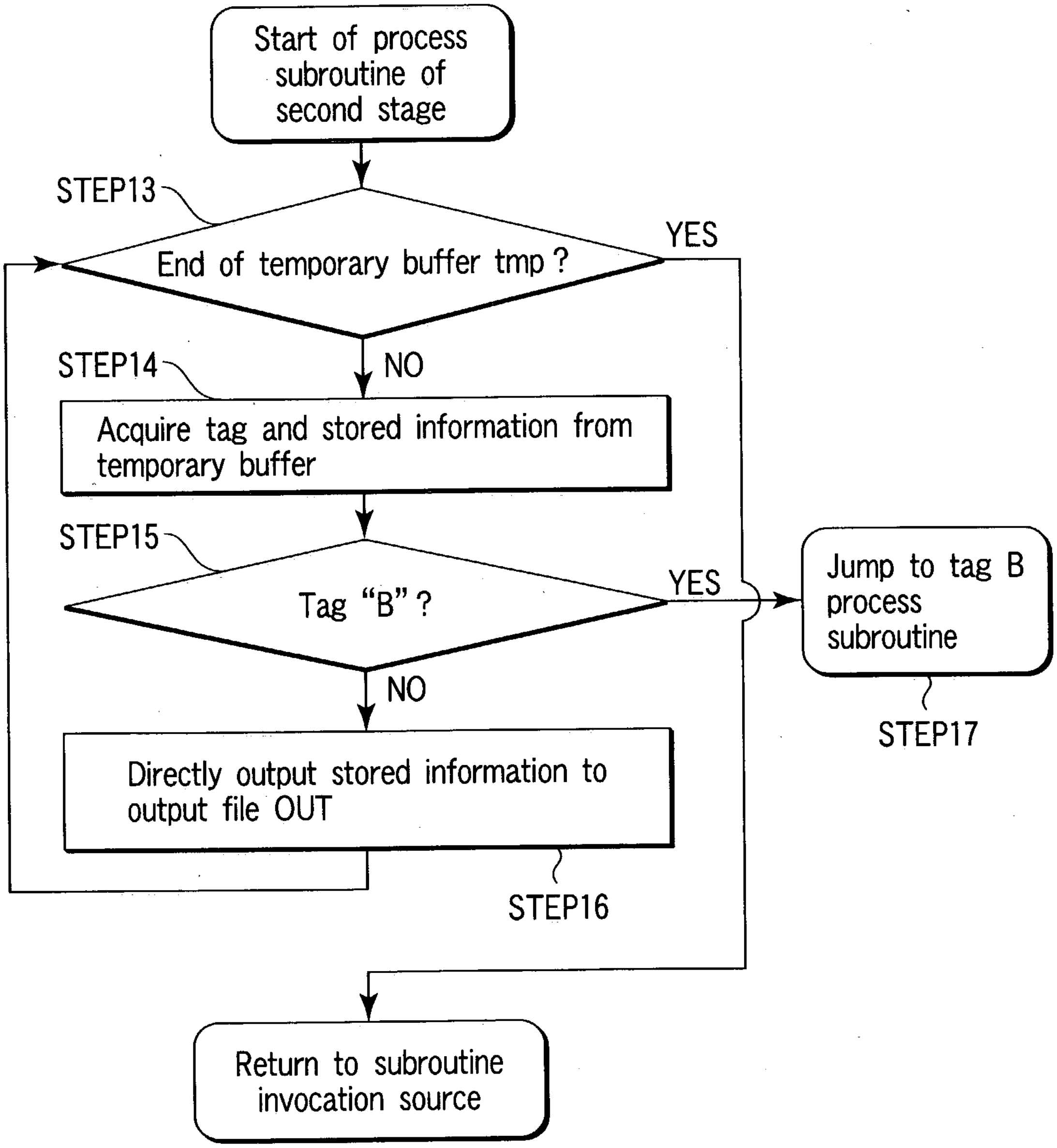

Start of process subroutine of second stage
STEP13
End of temporary buffer tmp ?
YES
NO
STEP14
Acquire tag and stored information from temporary buffer
STEP15
Tag "B" ?
YES
Jump to tag B process subroutine
STEP17
NO
Directly output stored information to output file OUT
STEP16
Return to subroutine invocation source

FIG. 6

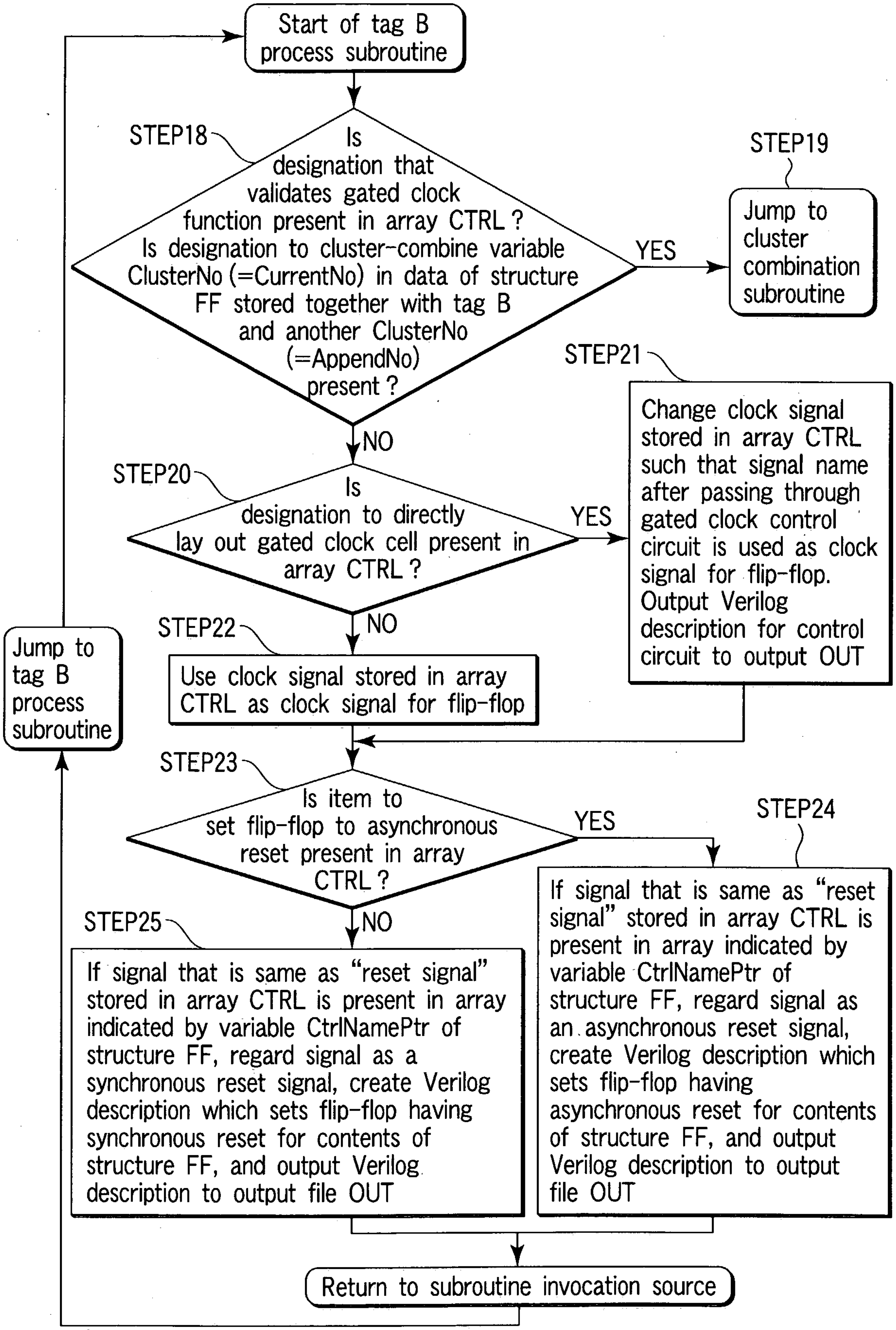
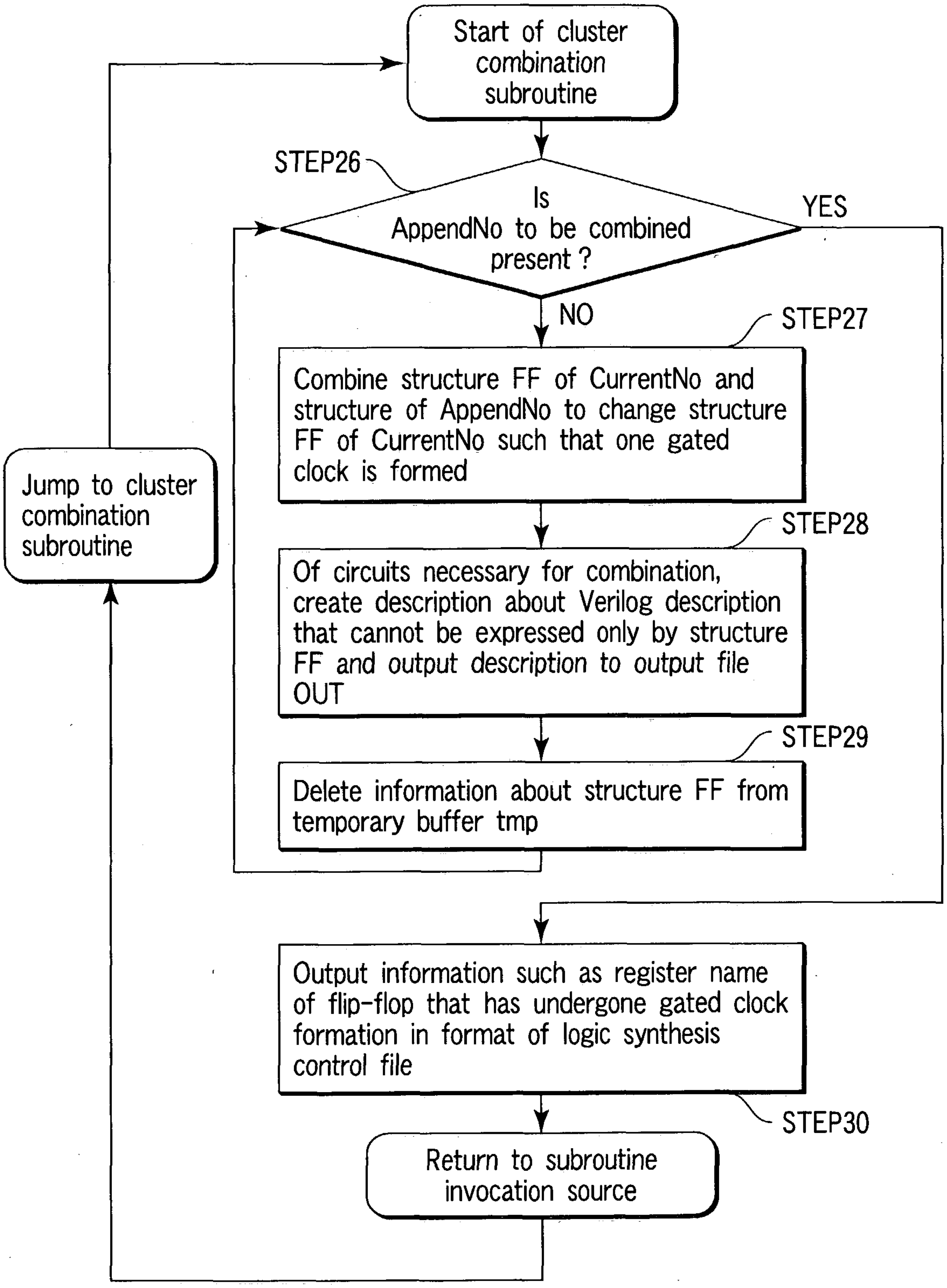
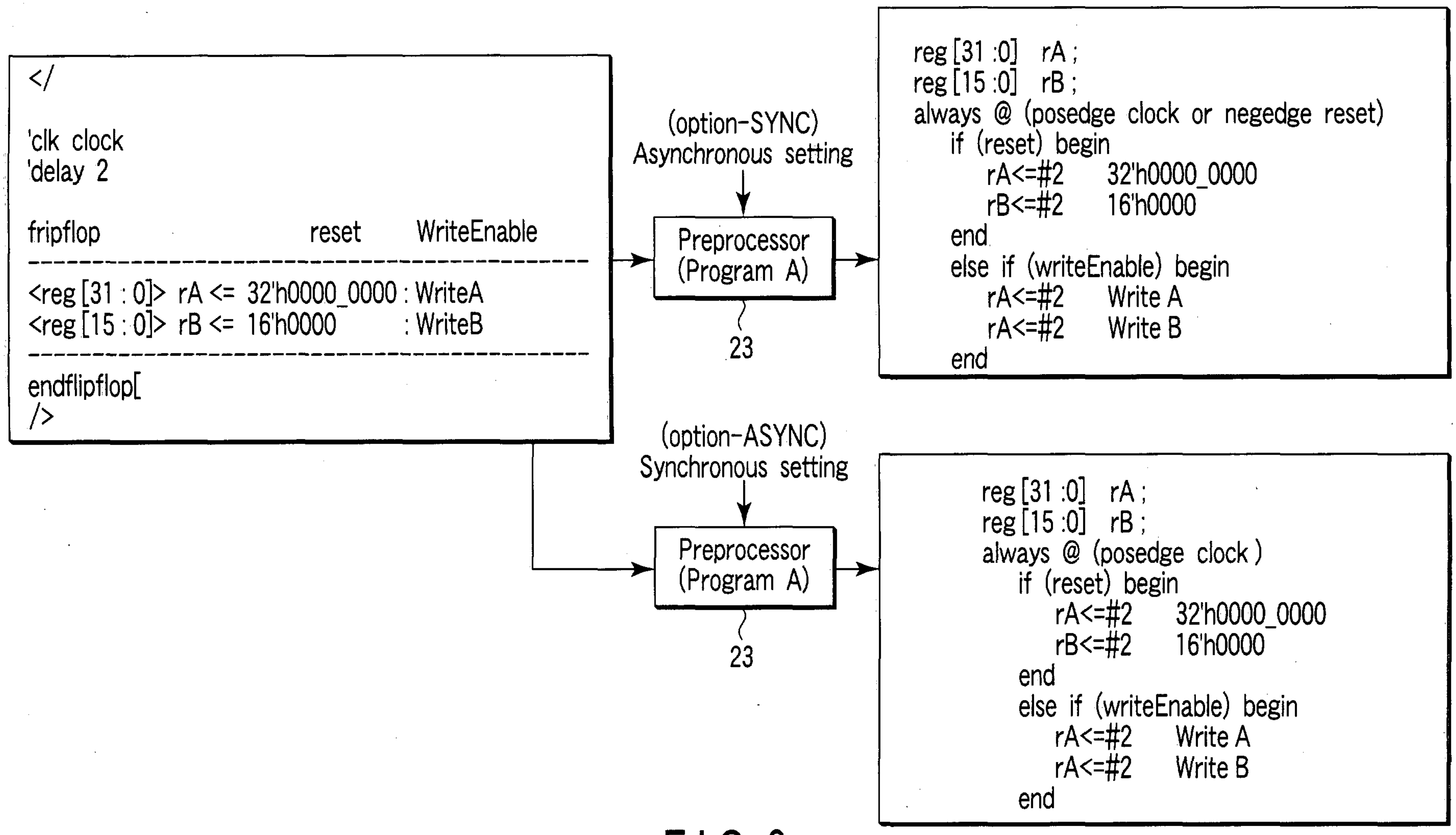

Start of tag B process subroutine
STEP18
Is designation that validates gated clock function present in array CTRL ? Is designation to cluster-combine variable ClusterNo (=CurrentNo) in data of structure FF stored together with tag B and another ClusterNo (=AppendNo) present ?
YES
STEP19
Jump to cluster combination subroutine
NO
STEP20
Is designation to directly lay out gated clock cell present in array CTRL ?
YES
STEP21
Change clock signal stored in array CTRL such that signal name after passing through gated clock control circuit is used as clock signal for flip-flop. Output Verilog description for control circuit to output OUT
NO
STEP22
Use clock signal stored in array CTRL as clock signal for flip-flop
Jump to tag B process subroutine
STEP23
Is item to set flip-flop to asynchronous reset present in array CTRL ?
YES
STEP24
If signal that is same as "reset signal" stored in array CTRL is present in array indicated by variable CtrlNamePtr of structure FF, regard signal as an asynchronous reset signal, create Verilog description which sets flip-flop having asynchronous reset for contents of structure FF, and output Verilog description to output file OUT
NO
STEP25
If signal that is same as "reset signal" stored in array CTRL is present in array indicated by variable CtrlNamePtr of structure FF, regard signal as a synchronous reset signal, create Verilog description which sets flip-flop having synchronous reset for contents of structure FF, and output Verilog description to output file OUT
Return to subroutine invocation source

FIG. 7


Start of cluster combination subroutine
STEP26
Is AppendNo to be combined present ?
YES
NO
STEP27
Combine structure FF of CurrentNo and structure of AppendNo to change structure FF of CurrentNo such that one gated clock is formed
STEP28
Of circuits necessary for combination, create description about Verilog description that cannot be expressed only by structure FF and output description to output file OUT
STEP29
Delete information about structure FF from temporary buffer tmp
Jump to cluster combination subroutine
Output information such as register name of flip-flop that has undergone gated clock formation in format of logic synthesis control file
STEP30
Return to subroutine invocation source

F I G. 8

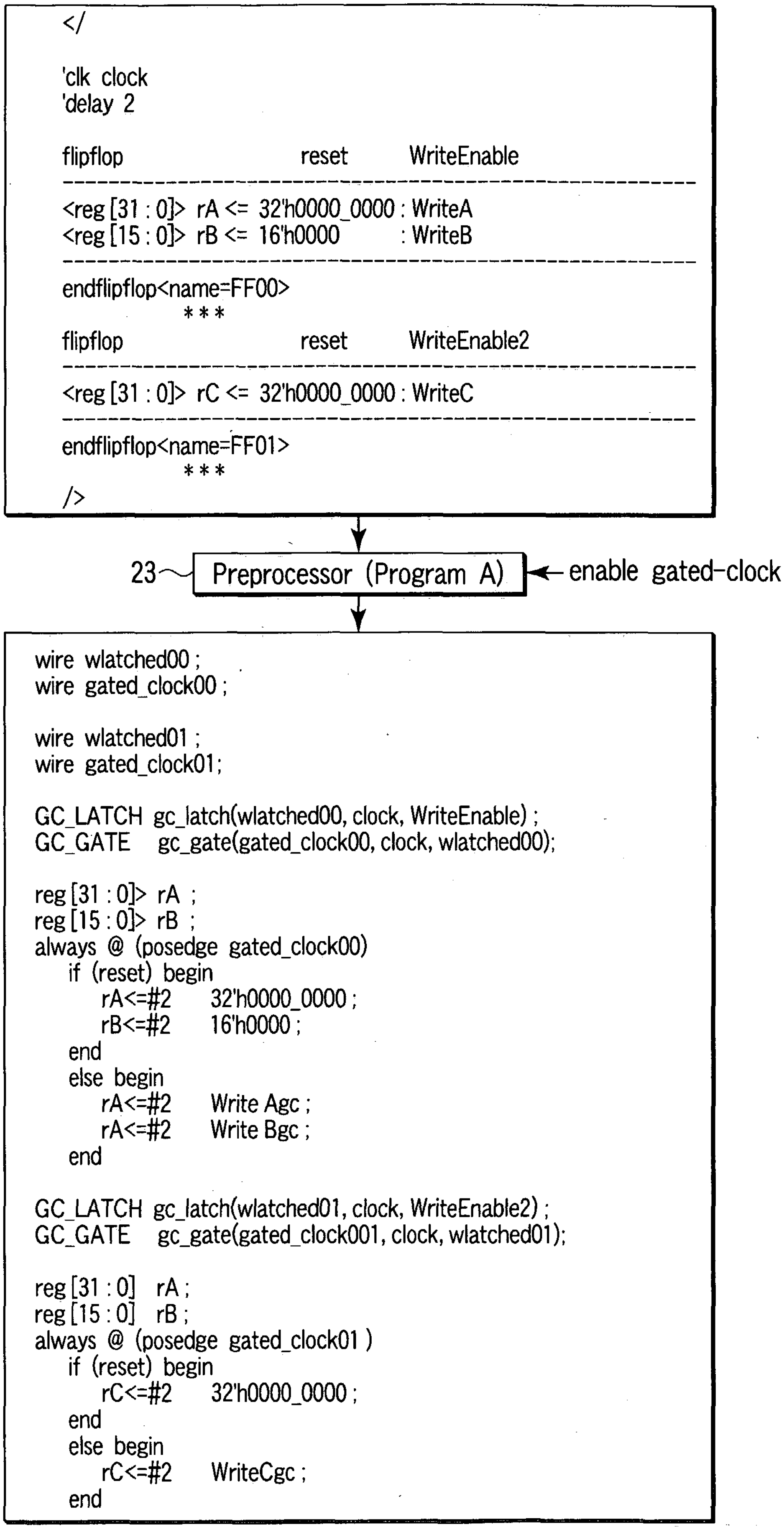


```
</

'clk  clock
'delay  2

fripflop                          reset        WriteEnable
---------------------------------------------------------
<reg [31 : 0]> rA <= 32'h0000_0000 : WriteA
<reg [15 : 0]> rB <= 16'h0000      : WriteB
---------------------------------------------------------
endflipflop[
/>
```

```
reg [31 :0]   rA ;
reg [15 :0]   rB ;
always @ (posedge clock or negedge reset)
    if (reset) begin
        rA<=#2     32'h0000_0000
        rB<=#2     16'h0000
    end
    else if (writeEnable) begin
        rA<=#2     Write A
        rA<=#2     Write B
    end
```

```
reg [31 :0]   rA ;
reg [15 :0]   rB ;
always @ (posedge clock )
    if (reset) begin
        rA<=#2     32'h0000_0000
        rB<=#2     16'h0000
    end
    else if (writeEnable) begin
        rA<=#2     Write A
        rA<=#2     Write B
    end
```

FIG. 9



```
</

'clk clock
'delay 2

flipflop                      reset        WriteEnable
------------------------------------------------------------
<reg [31 : 0]> rA <= 32'h0000_0000 : WriteA
<reg [15 : 0]> rB <= 16'h0000      : WriteB
------------------------------------------------------------
endflipflop<name=FF00>
             ***
flipflop                      reset        WriteEnable2
------------------------------------------------------------
<reg [31 : 0]> rC <= 32'h0000_0000 : WriteC
------------------------------------------------------------
endflipflop<name=FF01>
             ***
/>
```

```
wire wlatched00 ;
wire gated_clock00 ;

wire wlatched01 ;
wire gated_clock01;

GC_LATCH gc_latch(wlatched00, clock, WriteEnable) ;
GC_GATE    gc_gate(gated_clock00, clock, wlatched00);

reg [31 : 0]> rA ;
reg [15 : 0]> rB ;
always @ (posedge gated_clock00)
    if (reset) begin
        rA<=#2     32'h0000_0000 ;
        rB<=#2     16'h0000 ;
    end
    else begin
        rA<=#2     Write Agc ;
        rA<=#2     Write Bgc ;
    end

GC_LATCH gc_latch(wlatched01, clock, WriteEnable2) ;
GC_GATE    gc_gate(gated_clock001, clock, wlatched01);

reg [31 : 0]   rA ;
reg [15 : 0]   rB ;
always @ (posedge gated_clock01 )
    if (reset) begin
        rC<=#2     32'h0000_0000 ;
    end
    else begin
        rC<=#2     WriteCgc ;
    end
```

FIG. 10

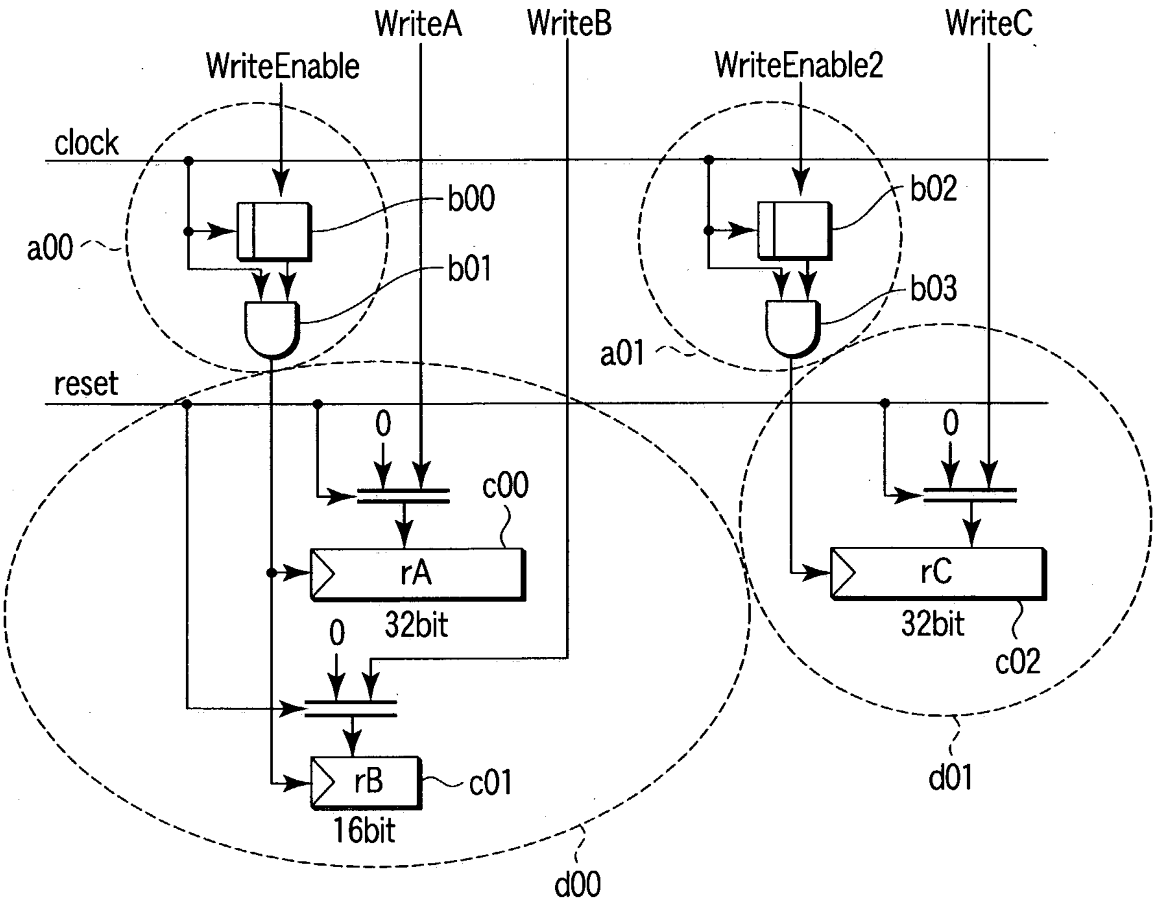
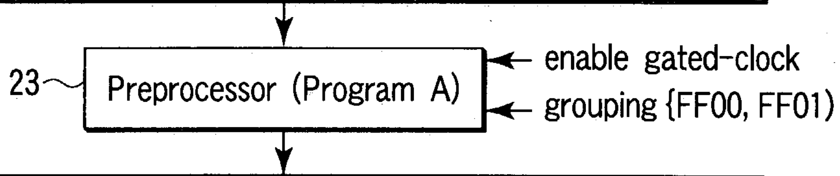

WriteA
WriteB
WriteC
WriteEnable
WriteEnable2
clock
b00
b02
a00
b01
b03
a01
reset
0
0
c00
rA
rC
32bit
32bit
c02
0
rB
c01
16bit
d01
d00

FIG. 11

```
</

'clk clock
'delay 2

flipflop                    reset        WriteEnable
--------------------------------------------------------
<reg [31 : 0]> rA <=  32'h0000_0000 : WriteA
<reg [15 : 0]> rB <=  16'h0000      : WriteB
--------------------------------------------------------
endflipflop<name=FF00>
           ***
flipflop                    reset        WriteEnable2
--------------------------------------------------------
<reg [31 : 0]> rC <=  32'h0000_0000 : WriteC
--------------------------------------------------------
endflipflop<name=FF01>
           ***
/>
```



```
wire [31 : 0] WriteAgc ;
wire [15 : 0] WriteBgc ;
wire [31 : 0] WriteCgc ;
wire gc_enable00 ;
wire wlatched00 ;
wire gated_clock00 ;

assign WriteAgc = WriteEnable   ? WriteA ; rA ;
assign WriteBgc = WriteEnable   ? WriteB ; rB ;
assign WriteCgc = WriteEnable2  ? WriteC ; rC ;

assign gc_enable00  = WriteEnable  |  WriteEnable2 ;

GC_LATCH gc_latch(wlatched00, clock, gc_enble00) ;
GC_GATE  gc_gate (gated_clock00, clock, wlatched00) ;

reg [31 : 0] rA ;
reg [15 : 0] rB ;
always @ (posedge gated_clock00)
   if (reset) begin
      rA  <= #2    32'h0000_0000 ;
      rB  <= #2    16'h0000 ;
      rC  <= #2    32'h0000_0000 ;
   end
   else if (WriteEnable) begin
       rA  <= #2 WriteAgc ;
       rB  <= #2 WriteBgc ;
       rC  <= #2 WriteCgc ;
   end
```

F I G. 12

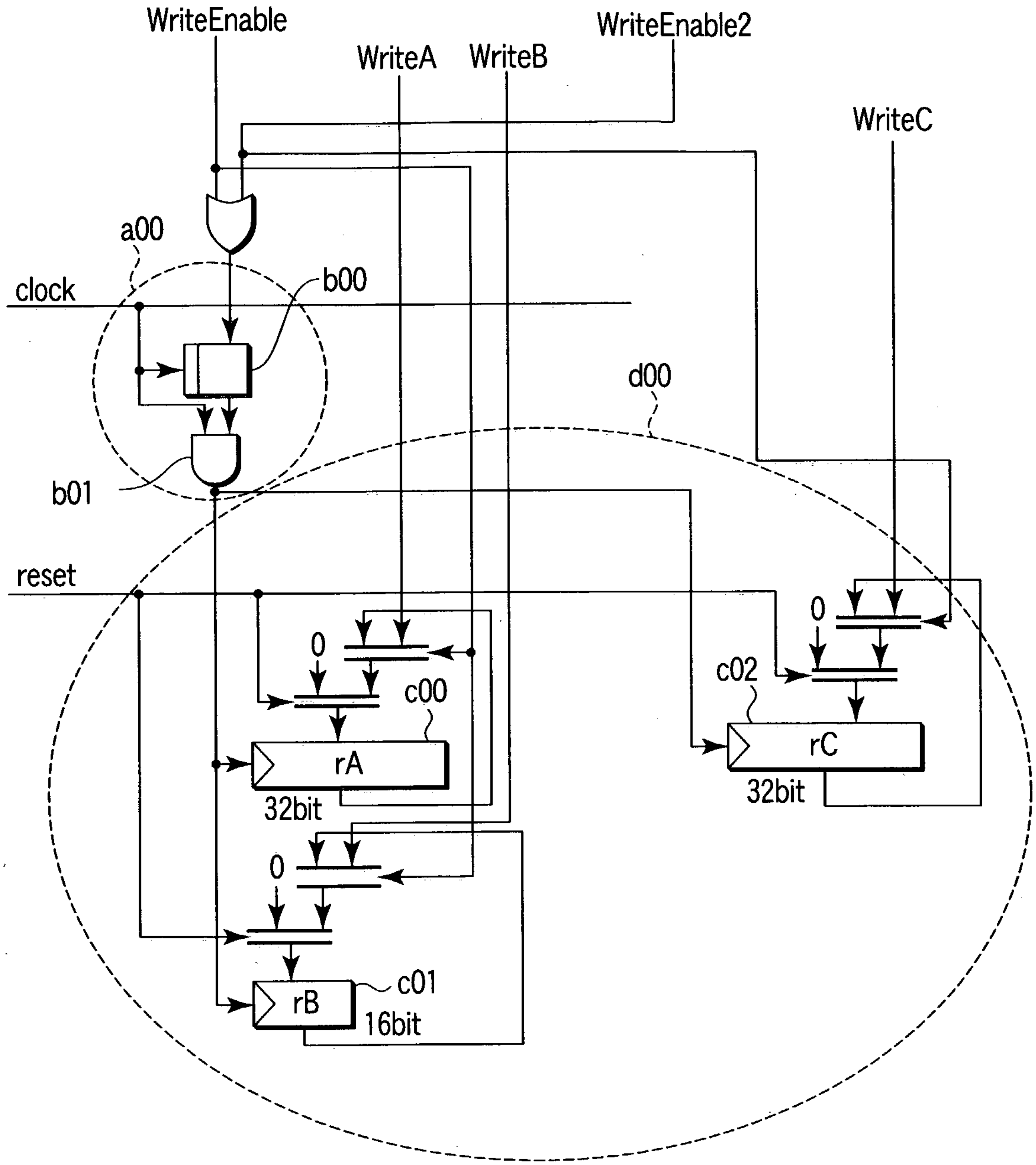

WriteEnable
WriteA
WriteB
WriteEnable2
WriteC
a00
clock
b00
d00
b01
reset
0
c00
rA
32bit
0
c02
rC
32bit
0
0
c01
rB
16bit

FIG. 13